

ArrayLists

Warm-Up

Think-Pair-Share

- Think about our use of arrays
- Write down some pros and cons of arrays




Review

- An **array** is a block of memory that allows us to store a list of related data
- **arrays** could store primitive data (**int**, **double**, **boolean**) and references to objects (**String**, any objects you make from a class)

```
int[ ] array = {10, 20, 30};  
String[] words = {"hello", "cat", "dog"};
```

- **Array limitation** - the size of the array was set in stone on initialization
 - difficult to add more to the end of the array, insert an element inside the array, and delete elements from the array

ArrayLists

- **ArrayLists** are resizable arrays that can only hold objects. These are sometimes referred to as lists (especially in other programming languages).
 - **ArrayLists** must hold object data - it can not hold primitive data (`int`, `double` and `boolean`) - For `int`, we use `Integer`. For `double`, we use `Double`. For `boolean`, we use `Boolean`.
 - **ArrayLists** have a number of methods that can be used to add, insert, delete and reorganize the data stored in them
- 

Declaring and Initializing ArrayLists


- to be able to use an **ArrayList**, we need to import the library that provides access to the **ArrayList** class and its methods
 - at the top of your program, need to have the statement

```
import java.util.ArrayList
```

- the syntax to declare and initialize an array is as follows:
 - `ArrayList<Type> name = new ArrayList<Type>();`

```
import java.util.ArrayList; // you may type java.util.* but it is not recommended
```

```
//declaring and initializing an ArrayList that will hold integers  
ArrayList<Integer> word = new ArrayList<Integer>();
```



Common methods

1. `void add(int index, TYPE o)`
2. `boolean add(TYPE o)`
3. `TYPE get(int index)`
4. `TYPE remove(int index)`
5. `boolean remove(TYPE value)`
6. `TYPE set(int index, TYPE o)`
7. `int size()`



Comparing Arrays and Lists

When to use a List or an Array?

Use an array when you want to store several items of the same type and you know how many items will be in the array and the items in the array won't change in order or number. Use a list when you want to store several items of the same type or not and you don't know how many items you will need in the list or when you want to remove items from the list or add items to the list.



Comparing Arrays and ArrayLists

Array

ArrayList

Declare

```
int[ ] highScores = null;  
String[ ] names = null;
```

```
ArrayList<Integer> highScoreList = null;  
ArrayList<String> nameList = null;
```

Create

```
int[ ] highScores = new int[5];
```

```
ArrayList<Integer> highScoreList = new  
ArrayList<Integer>();
```

Setting the value at an
index

```
highScores[0] = 80;
```

```
highScoreList.set(0,80);
```

Getting the value at an
index

```
int score = highScores[0];
```

```
int score = highScoreList.get(0);
```

Getting the number of
items

```
System.out.println(highScores.length);
```

```
System.out.println(highScoreList.size());
```



Adding Elements to an ArrayList

- Requires the syntax `ArrayListName.add(value);`

```
ArrayList<String> names = new ArrayList<String>();  
names.add("Hume");  
names.add("Locke");  
names.add("Hegel");  
System.out.println(names);
```

Output:
[Hume, Locke, Hegel]

- Adding elements to an ArrayList is slightly different than placing elements in an array
- requires the syntax `ArrayListName.add(value);`

Traversing ArrayLists with Loops

1. Enhanced For Each Loop
2. For Loop
3. While Loop

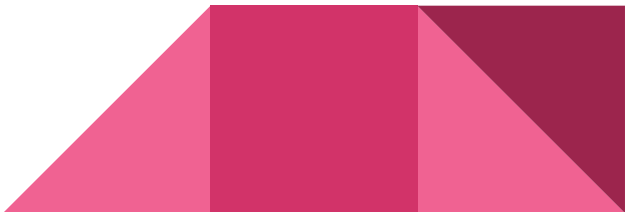


Enhanced For Each Loop

You can use an enhanced for-each loop to traverse through all of the items in a list, just like you do with an array.

```
import java.util.ArrayList;

public class EnhancedForEachLoop
{
    public static void main(String[] args)
    {
        ArrayList<Integer> myList = new ArrayList<Integer>();
        myList.add(50);
        myList.add(30);
        myList.add(20);
        int total = 0;
        for (Integer value: myList)
        {
            total += value;
        }
        System.out.println("Sum of all elements: " + total);
    }
}
```



For Loop

For Loop and While Loop process list elements using the index. The ArrayList index starts at 0 just like arrays, but instead of using the square brackets [] to access elements, you use the `get(index)` to get the value at the index and `set(index,value)` to set the element at an index to a new value. If you try to use an index that is outside of the range of 0 to the number of elements - 1 in an ArrayList, your code will throw an `ArrayIndexOutOfBoundsException`, just like in arrays.



For Loop

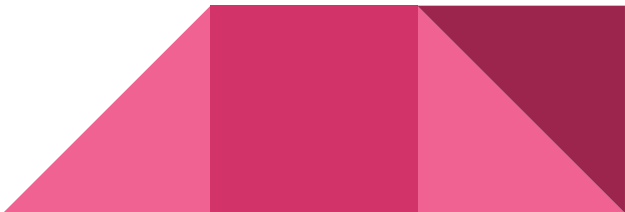
```
import java.util.ArrayList;

public class ForLoop
{
    public static void main(String[] args)
    {
        ArrayList<Integer> myList = new ArrayList<Integer>();
        myList.add(50);
        myList.add(30);
        myList.add(20);
        int total = 0;
        for (int i=0; i < myList.size(); i++)
        {
            total = total + myList.get(i);
        }
        System.out.println(total);
    }
}
```

While Loop

```
import java.util.ArrayList;
import java.lang.IndexOutOfBoundsException;

public class WhileLoop
{
    public static void main(String[] args)
    {
        ArrayList<Integer> myList = new ArrayList<Integer>();
        myList.add(50);
        myList.add(30);
        myList.add(20);
        int total = 0;
        int i = 0;
        try {
            while (i <= myList.size())
            {
                total = total + myList.get(i);
                i ++;
            }
        } catch (IndexOutOfBoundsException e){
            System.out.println("Error: index out of bounds");
        }
        System.out.println(total);
    }
}
```



Warning!!!

- Be careful when you remove items from a list as you loop through it. Remember that removing an item from a list will shift the remaining items to the left.
- Do not use the enhanced for each loop if you want to add or remove elements when traversing a list because it will throw a `ConcurrentModificationException` error.



Assume that `nums` is an `ArrayList` with these values `[0, 0, 4, 2, 5, 0, 3, 0]`. What will `nums` contain after executing the following code:

```
public static void numQuest(ArrayList<Integer> nums)
{
    int k = 0;
    int zero = 0;
    while (k < nums.size())
    {
        if (nums.get(k).equals(zero))
            nums.remove(k);
        k++;
    }
    System.out.println(nums);
}
```



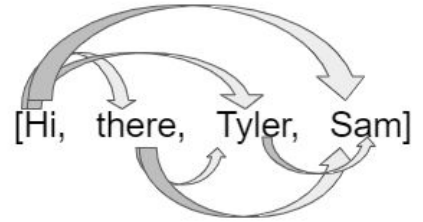
ArrayList of any Object

You can put any kind of Objects into an ArrayList. For example, [here](#) is an ArrayList of Students.



Practice

1. You are given a class called WordPair that can store pairs of words. Create an ArrayList of WordPair Objects.
2. a) In the class WordPairsList, write the constructor which takes the array of words and pairs them up as shown in the figure. You will need nested loops to pair each element with the rest of the elements in the list. Here is the pseudocode.
 - Initialize the allPairs list to an empty ArrayList of WordPair objects.
 - Loop through the words array for the first word in the word pair (for loop from index $i = 0$ to $\text{length}-1$)
 - Loop through the rest of the word array starting from index $i+1$ for the second word in the word pair (for loop from index $j = i+1$ to length)
 - Add the new WordPair formed from the i th word and the j th word to the allPairs ArrayList.




Pairs Created:
[Hi,there], [Hi,Tyler], [Hi,Sam],
[there,Tyler], [there,Sam],
[Tyler, Sam]

Practice

2. b) Write a method called `numMatches()` that counts and returns the number of pairs where the first word is the same as the second word. For example, if the word array is `["hi","bye","hi"]`, the pairs generated would be `["hi","bye"]`, `["hi","hi"]`, and `["bye","hi"]`. In the second pair `["hi","hi"]`, the first word is the same as the second word, so `numMatches()` would return 1.

For this method, you will need a loop that goes through the `ArrayList allPairs` and for each `WordPair` in `allPairs`, it checks to see if its first word (using the `getFirst()` method) equals the second word (using the `getSecond()` method). If there is a match, it increments a counter which it returns at the end of the method. To test this method, add another "there" into the words array and then uncomment the call to `numMatches()`.



Resources

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html>

